Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 41 (2012) 771 – 776

Engineering Procedia

www.elsevier.com/locate/procedia

International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012)

Low Cost Parallel Processing System for Image Processing Applications

Z. A. Ahmad[a], M. Elshaikh[b], C. M. Nor[c], M. S. Mustafa[d], M. A. Romli[e], M. F. Jamlos[f]

[a]zahari@unimap.edu.my,[b]elshaikh@unimap.edu.my,[c]cmnor@unimap.edu.my,
[d]sani@unimap.edu.my,[e]asmi@unimap.edu.my, [f]faizaljamlos@unimap.edu.my
[a,b,c,d,e]School of Computer and Communication Engineering,
Universiti Malaysia Perlis,
No.12 &14, Jalan Satu, Taman Seberang Jaya Fasa 3, Seberang Ramai,
02000 Kuala Perlis.

Abstract

The rapid grows in the image processing systems complexity, and the complicated nature of its new technologies system design, had created the need for a high speed embedded processing units at low costs. This paper proposes a new implementation method of low cost parallel processing system for image processing applications. The proposed method is implied two stages. The first stage focused in the implementation of the hardware of parallel processors, where two ARM technology processors are used for its considerably low cost, and as standard technology for processors. This stage also concerned about the overall system firmware requirements and system integrations. The second part of the work is to design the tasks parallelism between processors, and the overall system display. The complete system is tested for a real image processing data, and compared with a single processor system. The obtained results show that, the new system is significantly improved the processing in term of speed.

Keywords: Parallel processing, ARM processor, image processing.

Nomenclature

SISD	single instruction single data
MISD	multiple instruction single data
SIMD	single instruction multiple data
MIMD	multiple instruction multiple data
ARM	32 bit RISC processor
DFT	Discrete Fourier Transform

1. Introduction

Distributed computing technology is witnessed a radical development lastly, for it numerous applications in wide range of scientific and economic problems that need a large computer power for their solution. Technically speaking, distributed

* Corresponding author. Tel.: +6 012-492-9993; fax: +6 04-985-1654.
E-mail address: zahari@unimap.edu.my

1877-7058 © 2012 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.
doi:10.1016/j.proeng.2012.07.242

computing can be deployed with two different approaches. The first approach is by the use of power workstations connected in a network platform. This approach can be used in applications that require no mobility. The second approach is to implement a multiprocessing unit in a single device. The last approach is considered a good solution for application where the size is a matter such as some of image processing applications.

The race against speed of processing in computing has been intensely studied by many researchers lastly [1][2]. A general concern in parallel processing researches is to develop a high speed processing unit by deploying multi processor units, working together to execute tasks for the same program simultaneously [4]. In the literature once can find that many efforts had been made to increase the speed of a single processor [6]. Traditionally, the approaches to get tasks accomplished faster have been generalized on the design of single processors. There are efforts to increase the speed of a single processor by increasing memory size so that addressing can be directly and faster. Other approaches are used to make the processor more powerful by increasing the basic word length and computational precision [3][4].

The most popular approach is making the processor operate in high speed using micron-etching technology processor, putting more transistors in a chip and couple the processor with larger and faster communication pathways [5]. However those efforts are now reaching the limit. The smallest possible size of transistors has been produced; processor operating clock frequency is now about at the ceiling and cost of making a higher speed processor gone ridiculously high. The best alternative at the present time is to use the existing matured processors which operate at a fairly fast clock frequency and design them to operate in parallel; such approach optimizes the overall system cost, and achieves the desired high speed. This paper proposed a new parallel processing method for image processing application and its implementation. Technically speaking, image processing application implies the use of matrix, and applying a mathematical operation in image matrix. The fundamental about image processing is used to break down these matrixes into blocks and apply the operation in block bases, where different processors can be utilized to perform the desired computation. In this paper, two ARM processors are attached to a shared input memory, where input images are stored; a pre-image break down into smaller blocks is to be performed. This image break-down enables a huge size image to be processed in different processors simultaneously. The complete system is tested for a real image processing data, and compared with a single processor system. A comparison between the proposed system and traditional single processor is performed in terms of the processing delay.

The rest of this paper is organized as follow; the next section gives a brief idea about the parallel processing architecture, and the Flynn taxonomy classification of computer architecture. In section III, a description of the methods for the proposed approach is illustrated. Section IV describes the proposed system implementation and design, and it also shows the obtained results and its discussion. Section V concludes this paper.

2. Parallel processing architecture

Computer architecture can be classified into different types according to the execution mode. Based on the Flynn taxonomy classification of computer architecture as shown in Fig 1, the two categories of computer architecture are Instruction and Data, can take either Single or Multiple values [4]. Therefore, as per the realistic in computing world, there are basically three basic types of computer system.

The three types of computer system are Single Instruction Single data (SISD), Single Instruction and Multiple Data (SIMD) and Multiple Instruction and Multiple Data (MIMD).

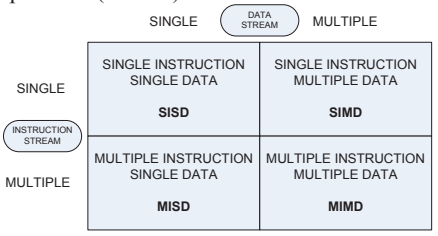

Fig. 1. Flynn taxonomy classification of computer architecture.

Parallelism can be exploited in Single Instruction Single data (SISD), Single Instruction and Multiple Data (SIMD) and Multiple Instruction and Multiple Data(MIMD) modes of execution based on two principles. Firstly, parallel implementation by overlapping operation in time-temporal parallelism. Secondly, parallel implementation by replicating resources in space which is called spatial parallelism.

There are four major types of parallel architecture. Firstly, the pipelining that interleaves the successive steps of executing an instruction or operation across multiple stages of pipeline unit [8]. This way of executing the instruction can then be overlapped when the pipeline is filled up. Conceptually, this implementation uses temporal parallelism. Secondly, processor array where multiple processors are connected together in an array [6]. Generalizing the pipeline mode into an

algorithm, then the algorithm can be broken into several steps where each step can be assigned to a processor in the array. This model of computation also called algorithmic parallelism. One example of such computational model is systolic array. In systolic array, the algorithm is decomposed to identical steps and mapped onto identical processing units. Thirdly, array processor architecture where multiple processors are connected together usually in a two-dimensional array and controlled by a single control unit [6]. The final architecture is multi-processor or multi-computer architecture [7]. Multi-processor architecture is formed by connecting multiple processors together for sharing a common memory. On the other hand, multi-computer is formed by connecting multi-computer together but not sharing the memory. The computers are interconnected via a high speed communication network. Each computer remains a full function computer in the system where they control their own CPU, memory and I/O's including the communication.

3. Methodology

The main objective of this paper is to design a parallel processing system for image processing application. Technically speaking, in image processing applications, the fundamental data structure used is matrix, so as to deploy a parallel processor in a matrix data structure, the idea is to break down these matrixes into smaller blocks, such that different processor can work simultaneously to process the image. Parallel processing of course relies on finding a way to partition the work to be done [9]. The blocks of operation are independently processed and later will yield the expected result when all results from each block are combined. So, the effectiveness of the parallel system is determined by the capability of breaking down the operations into small blocks. The blocks can be concurrently processed without giving any calculation errors. Figure 2, depicts the proposed system block diagram according to the sequence of the processes.

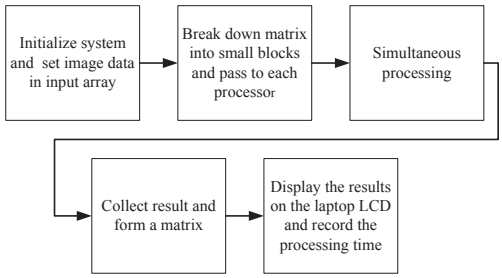

Fig. 2. General project process flow.

The computation for image processing is generally requires very long processing time. For example, the two dimensional Fourier Transform in image processing, image coding and image filtering applications requires substantial amount of processing time. One alternative for reducing the processing time is to compute the DFT on a parallel machine [3]. Evaluation of system performance is based on the speedup gain. It is one of the most common metrics used in the performance evaluation of parallel applications [5]. Speedup indicates the effectiveness of an increase in the number of processes used [5].

4. System implementation and design

The work in this paper is divided into two steps. The first step is concerned on the hardware design and implementation, and second step focuses on the firmware implementations. The next subsections describe these design steps in detail.

4.1 System hardware

The LPC 2148 development is a very good board whereby integration can be done easily. It is because the pre-wiring on the board and sufficient breadboard for external wiring are provided. The important General Purpose Input/Output (GPIO) are pre-wired nicely and connected to a single core wire terminal connectors. They can be connected with single core wire easily. There are also LEDs and Switches ready connected to the GPIO pins. All we need to do are to identify the pin that those LEDs and Switches are connected and set the port for output for LED and input for switch in the firmware program and they are ready to be used.

In this system, GPIO0 and GPIO1 are used. The LED is used to indicate the status ready and the communication activity between the two boards. The communication mean used is parallel data transfer through general purpose I/O port. The port used is GPIO1 utilizing an eight bit data line. This communication requires some control signals. Three signals are used to control the data transfer back and forth. GPIO0 port is used for the signals. Figure 3 shows the schematics diagram of the

wiring connections for the two boards.

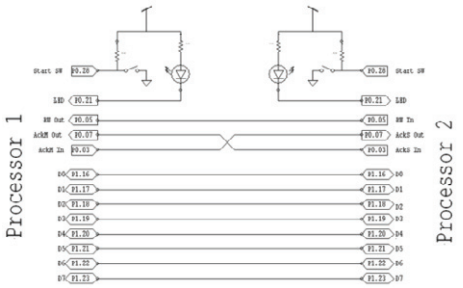

Fig. 3. Interface wiring diagram.

The board is also connected to laptop via serial communication port. This interfacing is used for sending resulting data to the laptop for display on the laptop's LCD. Serial COM port of the laptop is connected to serial UART port of board 1. The connection cable used is standard RS 232 cable with 9 pins D-Sub connectors male and female. Both communication links, ARM board to laptop pc and ARM board to ARM board, are using the shortest cable length possible to avoid signal lost. Furthermore, there are only two ARM processor boards and a laptop PC to be wired. Length of cable was not a big issue. Perhaps, the concern is that the communication will occur at a very fast rate; therefore we would anticipate that any signal loss due to communication cabling should be avoided.

4.2 Firmware

The firmware is developed using C ARM language. The development was efficiently done using the powerful Keil uVision3 Integrated Development Environment (IDE). The tool-chain setup in the IDE is Real View MDK-ARM Version 3.24. The IDE was also pre-setup with LPC2000 Flash programmer from Phillip Semiconductors. This interface was definitely helpful to boost up the speed of development. This setup allows the codes to be burned into the chip directly after finishing the compilation and HEX code generation.

The first firmware developed is to run the image processing using gray scale morphology processing method. The firmware is developed for the processor 1. There are two main functions to execute in the firmware. Firstly, the most important part of the firmware which is image processing routine where matrix operation and gray scale morphology method are used. For morphology, a 3 by 3 constructing element matrix is used to conditionally convolve with an image matrix. As a result of this convolution, image data on the first row, last row, first column and the last column will be washed out. This is happen when the constructing element sweep along the first three rows of the image matrix and altered only the data in the middle of the constructing element matrix. Therefore, the first row will be washed out along the way of the constructing element matrix run from left hand side to the right hand side on the image matrix. The same sequences apply to next row and until end of rows of image matrix. Finally, when all of rows are processed, the data is then sent to laptop to display on LCD.

Processing time is measured by using the VB capturing special data from processor 1. These data are used for start time and stop time of the processing. Before processor 1 start the matrix processing, it sends one character "@" to Laptop PC and interpreted by the VB program as a start point. The VB program is then stamp the start time using system time. After finishing the processing, processor 1 sends "end" string and this time the VB program interpret it as a stop signal. The VB program will stamp the time again from the system time and subtract it with the previous stamped time. The result is time different which equals the accumulative processing time of processor 1.

There are two tasks for the processor which the first task is a routine to process image by using morphology method. The second task is a routine to send the processed data to laptop via UART port.

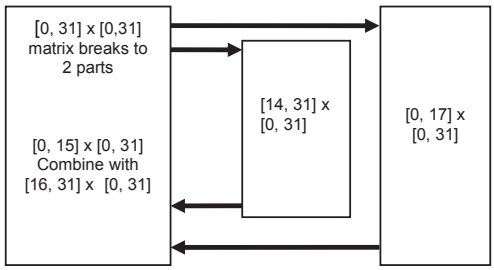

Fig. 4. Matrix breaking and data flow diagram.

On the other hand, the laptop has to run a program to accept the data through its serial port and use the data for constructing the image and display it on the LCD monitor. Therefore, a Visual Basic program is developed to accept data through serial port from Processor 1. The VB program is called "ARM Parallel Processing System Monitor". Its main functions are to display original image and processed image side by side so that comparison can be made. This program will also record the time of processing by marking the start time and the end time as described in the previous paragraph. Hence, the total processing time can be measured.

The second firmware is developed to interface the two processors and perform image processing simultaneously. The image matrix is divided into two groups and one group will be sent to the second processor to be processed simultaneously. After finish the processing, the second processor will return the data back to the first processor for image reconstruction. The flow of the process is shown in Fig 4. In order to do that, the communication has to be setup to allow data transfer back and forth between the two processors. It is decided to use parallel data transmission so that data transfer is done at the fastest rate. General purpose I/O port GPIO1 is used to transfer 8 bit data per cycle. A handshake protocol is developed to allow a correct data transmission at practically fastest rate. It is a big challenge to configure the protocol of parallel data transfer. This is due to few critical considerations which have to be satisfied. The first consideration is the transfer has to be fast enough to avoid transfer time delay. This delay cannot dominate the whole system processing time. Otherwise, we might not be able to gain improvement in the processing time. The second consideration is ARM processor does not have standard parallel communication port. This system is definitely requires a protocol that simple and efficient. We need to select and configure it manually. Eventually, the whole communication protocol is created uniquely. This parallel communication protocol might not be working if this protocol used in other type of processor. The communication protocol used in this system is described in the following paragraph.

Three control signals are assigned in the coding to control the data transfer. The control signals use port GPIO 0 port P0.03, P0.05 and P0.07 for both processor. P0.05 is the main signal to control data flow and controlled by Master which is P1. P2 acts as Slave where it can only read that signal at the same pin P0.05 but cannot write. Data will flow from P1 to P2 if P0.05 is low. On the other hand, data will flow from P2 to P1 whenever P0.05 is high. Only P1 can change P0.05 value. During the changeover of the bit P0.05, port GPIO1 pin P0.16 to P0.23 is set to the correct direction. Since the port is bidirectional, this task is easily set up. The port will set up to be either input or output based on the decided data flow. A careful setting has to be done during the changeover period to avoid any short to the port. Since the parallel data cable is connected directly from port to port(GPIO1) of processor 1 and processor 2, a short circuit can happen whenever both ports are set as output and one port is set to low while the other port set to high. This condition will allow high current flow from high output to the low input port. The consequences are the chip might get hot and if the current flow is sustain for long period of time, the chip might get burned and damaged.

Meanwhile P0.03 and P0.07 for both processors act as acknowledge input and output respectively. The acknowledge signal is very important in synchronizing the data transfer. The same signals are used for transfer of data from processor 1 to processor 2 and vice versa. If data flow from P1 to P2 (Master to Salve), P0.05 is low and whenever data setup is ready at port GPIO1, an acknowledge output (Ack Out) signal low will be sent to P2. P2 will receive it through its P0.03, the acknowledge input (Ack In). Upon receiving that signal, P2 will capture the GPIO1 port value into its memory. Then, upon finishing capturing port value, P2 will send acknowledge low signal through P0.07(Ack Out) to P1 indicating that the data has been read. Upon receiving this acknowledge signal through its acknowledge input (Ack In) pin, P1 is ready to send next data. Data vector is then points to the next data and set it at parallel output port (GPIO1). The sequence will repeat until all data is transferred. If data flow is from P2 to P1, the sequence is exactly the same but now GPIO1 direction is changed. P2 has to set it up to be output while P1 has to set it as input. P1 acts as master sending data to P2. When P1 want to receive data, it will set signal RW to high set its GPIO1 to be input. Data is then can be transmitted from Slave (P2) to Master (P1).

The morphology image processing is successfully accomplished in this project. The final processed image is shown in Fig 5. From there, we could observe that the image has been improved in term of clean, clarity and brightness. There is a room of improvement to this system where it can expand to higher resolution.

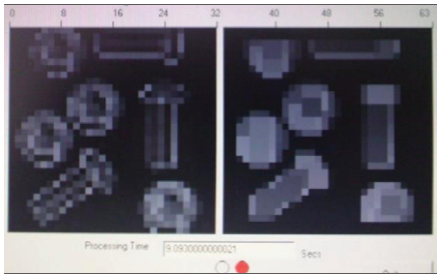

Fig. 5. Original image (left) and processed image (right).

The establishment of parallel communication protocol in this system is a big achievement. The system would not function efficiently without this communication protocol establishment. A very simple communication cabling is introduced. There is only three control signals used to control the communication. In addition, data bus is wired directly to the processor port without going into any external registers or multiplexer. Even current limiter resistors were not used. Finally, the communication link characteristic in this system is bi-directional.

Meanwhile, the expected speed up is achieved. However, it is not up to the theoretical value. There is a strong reason for that not to achieve. It is because of the communication delay that occurs during the data transfer between the two processors. After all, a quite distinct improvement is achieved in the processing time as compared to single processor. The speedup achieved is 28.31 %. The detail data is shown in the following Table 1.

5. Conclusion and further works

Speedup is the key measurement of a parallel processing system. In this paper two ARM processors are used to implement a parallel processing system for image processing. The system is consists of hardware and firmware implementations. The proposed system is successfully integrated and tested against a single processor. The loosely coupled processing parallel system in this research is successfully designed to achieve wall clock reduced by 28.31%. The detail results are shown in Table 1. More improvement in term of processing speed could be achieved by handling other system technical details such as, bus transfer rate, and matrix size.

There is a great concern on data transfer time in this system. It is the determinant for the performance or speed of completing tasks in a system with more than one processor. Therefore, one of further works to consider for this project is reducing the data transfer time between the processors. In this system only eight bit data bus is used but ARM has 32 bit data line [7] as provided by GPIO. If we employ the 32 bit data line, 4 bytes of data can be transferred in a cycle. This is definitely will improve transfer rate. Theoretically, the transfer time might improve to about 4 times faster than what we have now in this system. Hence, system performance would increase also.

Table 1.Processing time test data

Trial #	Processing Time, Sec		Delta	% Speedup
	Single Processor	Two Processor		
1	12.688	9.093	3.594	28.33
2	12.687	9.093	3.594	28.33
3	12.678	9.092	3.585	28.28
4	12.689	9.093	3.596	28.34
5	12.678	9.093	3.584	28.28
6	12.688	9.092	3.595	28.33
7	12.688	9.093	3.594	28.33
8	12.689	9.092	3.596	28.34
9	12.687	9.102	3.584	28.26
10	12.689	9.102	3.585	28.26
Ave.	**12.686**	**9.095**	**3.591**	**28.31**

The other concern is that matrix size whereby in this project only 32 by 32 matrix size was used. Convolution is done only twice for this matrix. So, it seems inadequate to test the system real performance. A variety of matrix sizes should be used to test the system and also, many other matrix operations shall be performed using this system. As long as the operation can be divided into small parts, this system should be able to process it. Furthermore, it should not be limited to matrix operation in image processing only.

References

[1] Matloff, Norman, "Introduction to parallel matrix operation*," University California at Davis*, May 20, 2008.
[2] Jin, Lin, "Parallel processing: exploring the architectures' and algorithms' close relation," *IEEE*, December 94/January 95.
[3] J. T. Rayfield and H.F. Harvey, "Implementation of 2-D DFT algorithm on a loosely-coupled parallel system," *IEEE*.
[4] Elmohamed, Saleh ,"Parallel processing concepts" *Cornell University,* October 25, 2002.
[5] S. Novak, J. Lamb, T. Green, L. Weldon, B. Fleisch, "Performance evaluation for a loosely coupled parallel processing environment," *Tulane University*.
[6] O.A McBryan, "Trends in Parallel Computing," *Colorado University at Boulder*, December 1990.
[7] "JX-2148 : LPC2148 ARM7-32 bit microcontroller education board. JX-2144 manual," *INEXGlobal*, June 5, 2006.
[8] B. Eric*, "*Implement parallel processing in your Java applications," http://www.devx.com/Java/Article/34289/0/page/1 April 10, 2007
[9] H. El-Rewini, and M. Abd-El-Barr, *Advanced Computer Architecture and Parallel Processing*, John Wiley, 2004.
[10] H. Kitano, "Massively Parallel Artificial Intelligent,"*Carnegie Mellon University*, December 2006.
[11] H. Yamana, T. Marushima, T. Hagiwara, and Y. Muraoka, "System architecture of parallel processing system," *In Proceedings of the 2nd Iinternational Conference on Supercomputing (St. Malo, France). J. Lenfant, Ed. ICS '88*, *New York*, pp.76-89.
[12] A. Goller, and F. Leberl, "Radar image processing with clusters of computers aerospace and electronic systems magazine," *IEEE Volume 24*, Issue 1, Jan. 2009, pp.18 – 22.